Práctica 1.2. TCP y NAT

Objetivos

En esta práctica estudiaremos el funcionamiento del protocolo TCP. Además, veremos algunos parámetros que permiten ajustar el comportamiento de las aplicaciones TCP. Finalmente, se verá cómo configurar NAT con `iptables`.

Activar el **portapapeles bidireccional** (menú Dispositivos) en las máquinas.

Usar la opción de Virtualbox (menú Ver) para realizar **capturas de pantalla**.

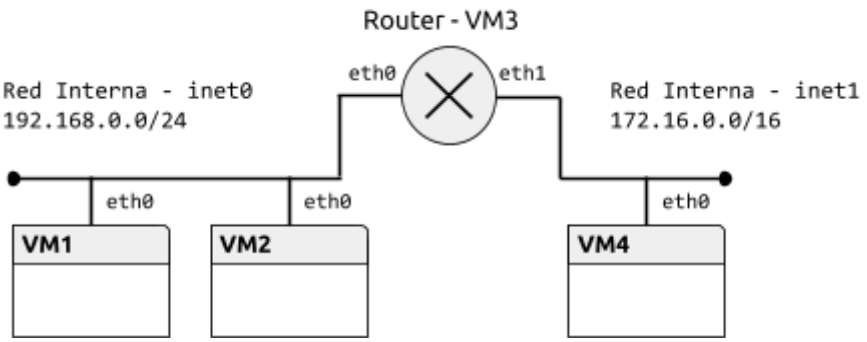
La contraseña del usuario `cursoredes` es `cursoredes`.

Contenidos

- Preparación del entorno para la práctica
- Estados de una conexión TCP
- Introducción a la seguridad en el protocolo TCP
- Opciones y parámetros TCP
- Traducción de direcciones (NAT) y reenvío de puertos (*port forwarding*)

Preparación del entorno para la práctica

Configuraremos la topología de red que se muestra en la siguiente figura, igual a la empleada en la práctica anterior.



Antes de crear el entorno **eliminar las máquinas virtuales de ASOR de VirtualBox**, junto con todos sus archivos. Después **importar el servidor** usando `/mnt/DiscoVMs/ASOR/ASOR-FE.ova`. Finalmente **crear la topología con `vtopol`**.

El contenido del fichero de configuración de la topología debe ser el siguiente:

```
netprefix inet
machine 1 0 0
machine 2 0 0
machine 3 0 0 1 1
machine 4 0 1
```

Finalmente, configurar la red de todas las máquinas de la red según la siguiente tabla. Después de configurar todas las máquinas, comprobar la conectividad con la orden `ping`.

Máquina	**Dirección IPv4**	**Comentarios**
VM1	`192.168.0.1/24`	Añadir Router como encaminador por defecto
VM2	`192.168.0.2/24`	Añadir Router como encaminador por defecto
Router - VM3	`192.168.0.3/24 (eth0)` `172.16.0.3/16 (eth1)`	Activar el *forwarding* de paquetes
VM4	`172.16.0.4/16`	Añadir Router como encaminador por defecto

Estados de una conexión TCP

En esta parte usaremos la herramienta Netcat, que permite leer y escribir en conexiones de red. Netcat es muy útil para investigar y depurar el comportamiento de la red en la capa de transporte, ya que permite especificar un gran número de los parámetros de la conexión. Además para ver el estado de las conexiones de red usaremos el comando `ss` (similar a `netstat`, pero más moderno y completo).

Ejercicio 1. Consultar las páginas de manual de `nc` y `ss`. En particular, consultar las siguientes opciones de `ss`: `-a`, `-l`, `-n`, `-t` y `-o`. Probar algunas de las opciones para ambos programas para familiarizarse con su comportamiento.

Ejercicio 2. (LISTEN) Abrir un servidor TCP en el puerto 7777 en VM1 usando el comando `nc -l 7777`. Comprobar el estado de la conexión en el servidor con el comando `ss -tln`. Abrir otro servidor en el puerto 7776 en VM1 usando el comando `nc -l 192.168.0.1 7776`. Observar la diferencia entre ambos servidores usando `ss`. Comprobar que no es posible la conexión desde VM1 con `localhost` como dirección destino usando el comando `nc localhost 7776`.

Adjuntar la salida del comando ss correspondiente a los servidores.

*En **VM1**:*

```
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN   0      100    127.0.0.1:25                *:*
LISTEN   0      10     *:7777                      *:*
LISTEN   0      128    *:111                       *:*
LISTEN   0      128    *:22                        *:*
LISTEN   0      128    127.0.0.1:631               *:*
LISTEN   0      100    ::1:25                      :::*
LISTEN   0      10     :::7777                     :::*
LISTEN   0      128    :::111                      :::*
LISTEN   0      128    :::22                       :::*
LISTEN   0      128    ::1:631                     :::*
```

*En **VM1**:*

```
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN   0      10     192.168.0.1:7776            *:*
```

*En **VM1**:*

```
nc localhost 7776
Ncat: Connection refused.
```

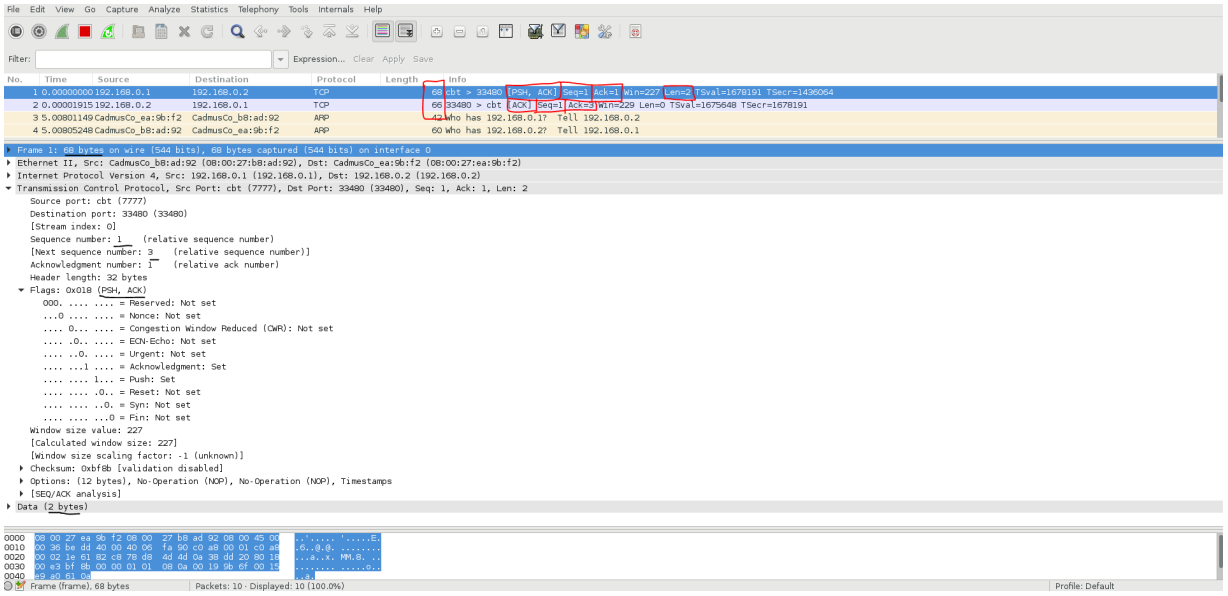
Ejercicio 3. (ESTABLISHED) En VM2, iniciar una conexión cliente al primer servidor arrancado en el ejercicio anterior usando el comando `nc 192.168.0.1 7777`.

- Comprobar el estado de la conexión e identificar los parámetros (dirección IP y puerto) con el comando `ss -tn`.
- Iniciar una captura con Wireshark. Intercambiar un único carácter con el cliente y observar los mensajes intercambiados (especialmente los números de secuencia, confirmación y flags TCP) y determinar cuántos bytes (y número de mensajes) han sido necesarios.

Adjuntar la salida del comando ss correspondiente a la conexión y una captura de pantalla de Wireshark.

*En **VM2**:(ss -tn)*

State	*Recv-Q*	*Send-Q*	*Local Address:Port*	*Peer Address:Port*
ESTAB	*0*	*0*	*192.168.0.2:33480*	*192.168.0.1:7777*



Al mandar un carácter del servidor al cliente vemos en las mensajes:
Longitud 68 bits primero y 66 el segundo que no contiene datos
Seq 1 para ambos y Ack 1 y 3 para el primero y segundo respectivamente
Los flags activados son PSH ACK para el primero y ACK para el segundo

Ejercicio 4. (TIME-WAIT) Cerrar la conexión en el cliente (con `Ctrl+C`) y comprobar el estado de la conexión usando `ss -tan`. Usar la opción `-o` de `ss` para observar el valor del temporizador TIME-WAIT.

Adjuntar la salida del comando ss correspondiente a la conexión.
*En **VM2**: (ss -tan)*

State	*Recv-Q*	*Send-Q*	*Local Address:Port*	*Peer Address:Port*
TIME-WAIT	*0*	*0*	*192.168.0.2:33480*	*192.168.0.1:7777*

La opción -o
timer:(timewait,1.231ms,0)

Ejercicio 5. (SYN-SENT y SYN-RECV) El comando `iptables` permite filtrar paquetes según los flags TCP del segmento con la opción `--tcp-flags` (consultar la página de manual `iptables-extensions`).

Usando esta opción:

- Fijar una regla en el servidor (VM1) que bloquee un mensaje del acuerdo TCP de forma que el cliente (VM2) se quede en el estado SYN-SENT. Comprobar el resultado con `ss -tan` en el cliente.
- Borrar la regla anterior y fijar otra en el cliente (VM2) que bloquee un mensaje del acuerdo TCP de forma que el servidor se quede en el estado SYN-RECV. Comprobar el resultado con `ss -tan` en el servidor. Además, esta regla debe dejar al servidor también en el estado LAST-ACK después de cerrar la conexión en el cliente. Usar la opción `-o` de `ss` para determinar cuántas retransmisiones se realizan y con qué frecuencia. Borrar la regla al terminar.

Adjuntar los comandos iptables utilizados y la salida del comando ss correspondiente a las conexiones.

*En **VM1**:*
iptables -A INPUT -s 192.168.0.2 -p tcp --tcp-flags ALL SYN -j DROP (aplicamos una regla sobre VM2 que desactiva todos los flags salvo el SYN)

nc -l 7777

*En **VM2**:*
nc 192.168.0.1 7777
ss -tan

State Recv-Q Send-Q Local Address:Port Peer Address:Port
SYN-SENT 0 1 192.168.0.2:33484 192.168.0.1:7777

*Para borrar la regla desde **VM1** iptables -D INPUT 1*

*En **VM2**:*
iptables -A OUTPUT -s 192.168.0.2 -p tcp --tcp-flags ALL ACK -j DROP

nc 192.168.0.1 7777

*En **VM1**:*

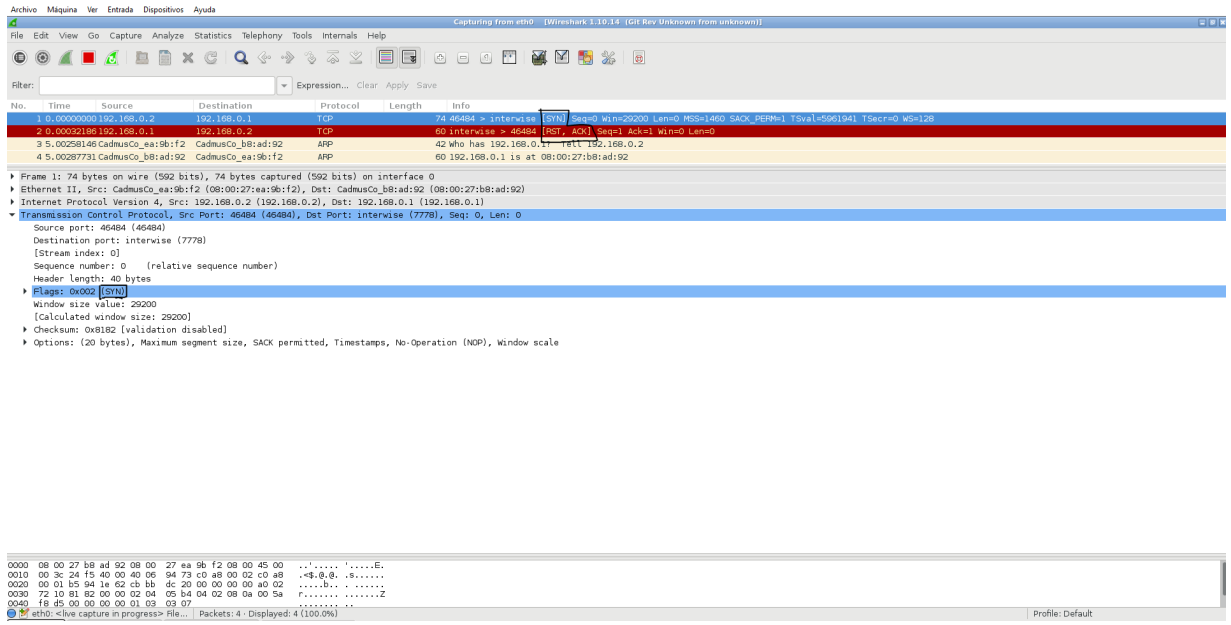
State Recv-Q Send-Q Local Address:Port Peer Address:Port
*LISTEN 0 10 *:7777 *:* timer:(keepalive,108ms,0)*
SYN-RECV** 0 0 192.168.0.1:7777 192.168.0.2:33496 **timer:(on,253ms,1)
SYN-RECV** 0 0 192.168.0.1:7777 192.168.0.2:33496 **timer:(on,6.998ms,3)
SYN-RECV** 0 0 192.168.0.1:7777 192.168.0.2:33496 **timer:(on,14sec,4)
SYN-RECV** 0 0 192.168.0.1:7777 192.168.0.2:33496 **timer:(on,31sec,5)

Observamos se mandan mensajes con frecuencia creciente (lo vemos haciendo ss -tan -o sucesivamente)
Si cortamos la conexión del cliente (Ctrl+C) aparece el estado Last-Ack
SYN-RECV 0 0 192.168.0.1:7777 192.168.0.2:33498 timer:(on,3.961ms,3)
***LAST-ACK** 0 1 192.168.0.1:7777 192.168.0.2:33498 timer:(on,35sec,0)*

Para borra la regla desde VM2 iptables -D OUTPUT 1

Ejercicio 6. Iniciar una captura con Wireshark. Intentar una conexión a un puerto cerrado del servidor (ej. 7778) y observar los mensajes TCP intercambiados, especialmente los flags TCP.

Adjuntar una captura de pantalla de Wireshark.
*En **VM2**:*
nc 192.168.0.1 7778
Ncat: Connection refused.



*Los flags que podemos observar sonen el primer mensaje **SYN** y en el segundo **RST** y **ACK***

Introducción a la seguridad en el protocolo TCP

Diferentes aspectos del protocolo TCP pueden aprovecharse para comprometer la seguridad del sistema. En este apartado vamos a estudiar dos: ataques DoS basados en TCP SYN *flood* y técnicas de exploración de puertos.

Ejercicio 7. El ataque TCP SYN *flood* consiste en saturar un servidor mediante el envío masivo de mensajes SYN.

- (Cliente VM2) Para evitar que el atacante responda con un mensaje RST (que liberaría la conexión), bloquear con `iptables` los mensajes SYN+ACK del servidor.
- (Cliente VM2) Usar el comando `hping3` (estudiar la página de manual) para enviar mensajes SYN al puerto 22 del servidor (`ssh`) lo más rápido posible (*flood*).
- (Servidor VM1) Estudiar el comportamiento de la máquina, en términos del número de paquetes recibidos. Comprobar si es posible la conexión al servicio `ssh` desde Router.

Repetir el ejercicio desactivando el mecanismo SYN *cookies* en el servidor con el comando `sysctl` (parámetro `net.ipv4.tcp_syncookies`).

Adjuntar los comandos iptables y hping3 utilizados. Describir el comportamiento de la máquina con y sin el mecanismo SYN cookies.
*En **VM2**:*
iptables -A INPUT -s 192.168.0.1 -p tcp --tcp-flag ALL SYN,ACK -j DROP

hping3 --flood -p 22 -S 192.168.0.1
HPING 192.168.0.1 (eth0 192.168.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

Al cerrar conexión: --- 192.168.0.1 hping statistic ---
***2618874** packets transmitted, 0 packets received, 100% packet loss*
round-trip min/avg/max = 0.0/0.0/0.0 ms

*En **VM3**: es posible la conexión*
nc 192.168.0.1 22
SSH-2.0-OpenSSH_7.4

*Para ver los paquetes en **VM1** Ifconfig eth0*
*RX packets **2629267** bytes 157763876 (150.4 MiB)*
TX packets 2629882 bytes 157804193 (150.4 MiB)

En VM1:
sysctl net.ipv4.tcp_syncookies=0

*En **VM2**:*

hping3 --flood -p 22 -S 192.168.0.1
HPING 192.168.0.1 (eth0 192.168.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

Al cerrar conexión: --- 192.168.0.1 hping statistic ---
***1778760** packets transmitted, 0 packets received, 100% packet loss*
round-trip min/avg/max = 0.0/0.0/0.0 ms

*En **VM3**: es posible la conexión*
nc 192.168.0.1 22
Ncat: Connection timed out.

*Para ver los paquetes en **VM1** Ifconfig eth0*
*RX packets **4410581** bytes 264642822 (252.3 MiB)* ***Incremento de 1781314 desde antes***
TX packets 2637217 bytes 158244352 (150.9 MiB)

Nota: Wireshark no debe estar activo cuando se envían paquetes lo más rápido posible (*flooding*).

Ejercicio 8. (Técnica CONNECT) Netcat permite explorar puertos usando la técnica CONNECT que intenta establecer una conexión a un puerto determinado. En función de la respuesta (SYN+ACK o RST), es posible determinar si hay un proceso escuchando.

- (Servidor VM1) Abrir un servidor en el puerto 7777.
- (Cliente VM2) Explorar, de uno en uno, el rango de puertos 7775-7780 usando nc, en este caso usar las opciones de exploración (-z) y de salida detallada (-v).
- Con ayuda de Wireshark, observar los paquetes intercambiados.

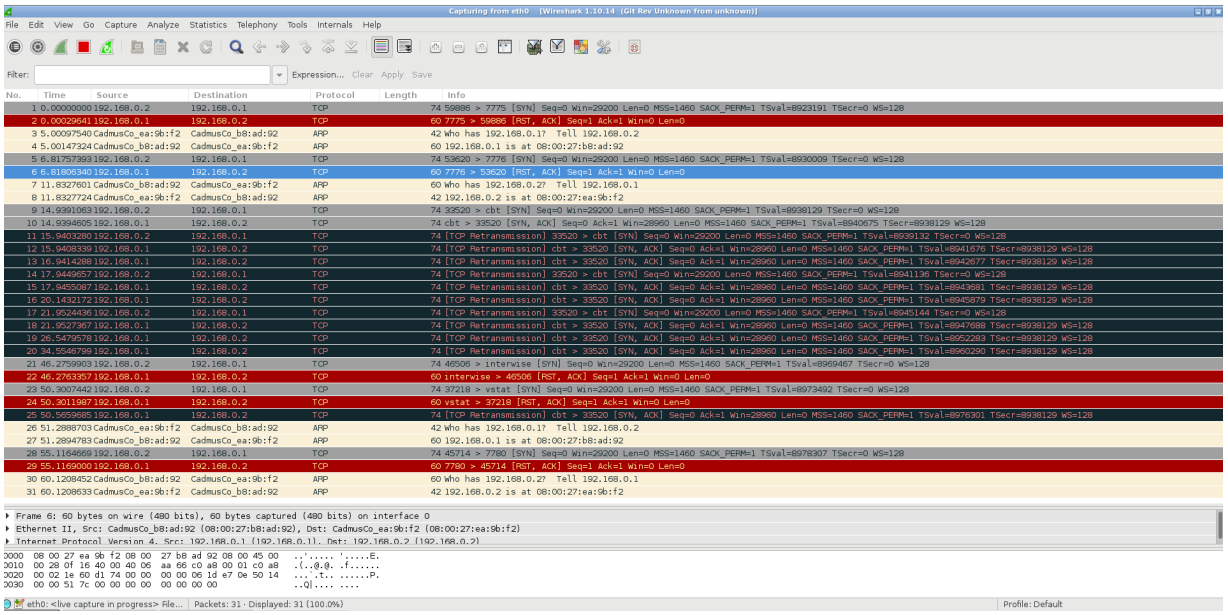
Adjuntar los comandos nc utilizados y su salida.
*En **VM1** nc -l 7777*
*En **VM2:***
nc -z -v 192.168.0.1 7775
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.

nc -z -v 192.168.0.1 7776
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused

nc -z -v 192.168.0.1 7777
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection timed out. (Desconozco por qué

nc -z -v 192.168.0.1 7778
Ncat: Version 7.50 (https://nmap.org/ncat)

Ncat: Connection refused.
nc -z -v 192.168.0.1 7779
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.
[nc -z -v 192.168.0.1 7780
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.



Opcional. La herramienta Nmap permite realizar diferentes tipos de exploración de puertos, que emplean estrategias más eficientes. Estas estrategias (SYN *stealth*, ACK *stealth*, FIN-ACK *stealth*…) se basan en el funcionamiento del protocolo TCP. Estudiar la página de manual de `nmap` (PORT SCANNING TECHNIQUES) y emplearlas para explorar los puertos del servidor. Comprobar con Wireshark los mensajes intercambiados.

Opciones y parámetros de TCP

El comportamiento de la conexión TCP se puede controlar con varias opciones que se incluyen en la cabecera en los mensajes SYN y que son configurables en el sistema operativo por medio de parámetros del kernel.

Ejercicio 9. Con ayuda del comando `sysctl` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten modificar algunas opciones de TCP:

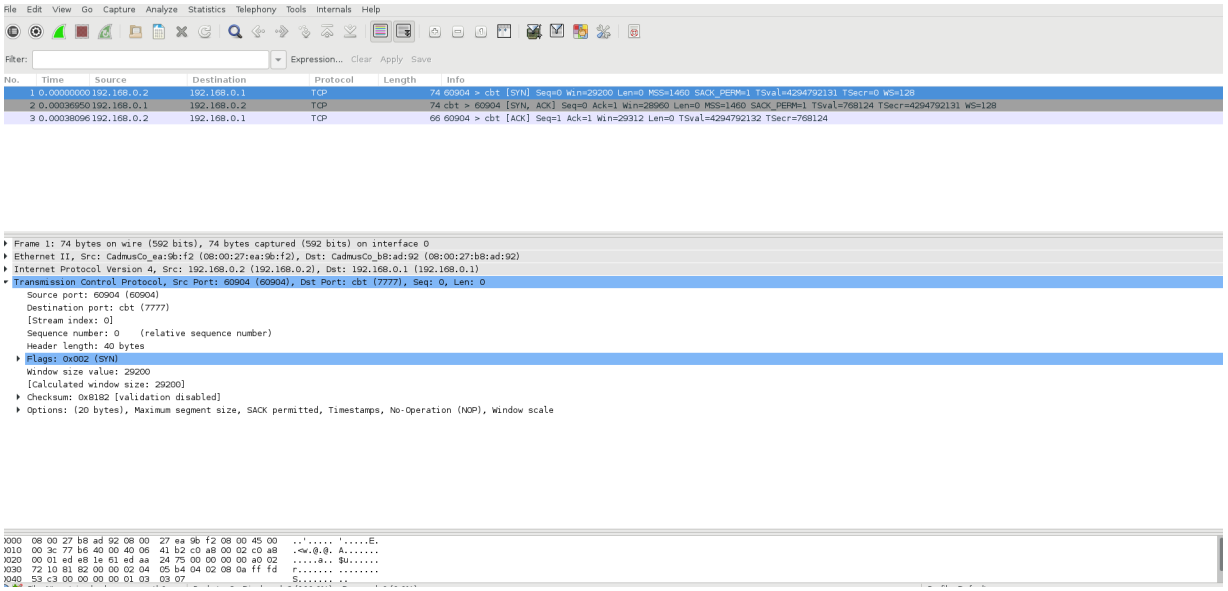
Parámetro del kernel	**Propósito**	**Valor por defecto**
`net.ipv4.tcp_window_scaling`	La opción tcp_window_scaling permite un tamaño de ventana superior a 65 K bytes mediante el uso de un factor de escala para multiplicar el valor del tamaño de la ventana. Este factor se establece en función del tamaño máximo del búfer de	1

	recepción utilizado por el socket TCP	
`net.ipv4.tcp_timestamps`	Se utiliza para aproximar el tiempo de actividad del host remoto y ayudar en nuevos ataques	1
`net.ipv4.tcp_sack`	Los SACK funcionan añadiendo a un paquete de confirmación duplicado una opción TCP que contiene un rango de datos no contiguos recibidos	1

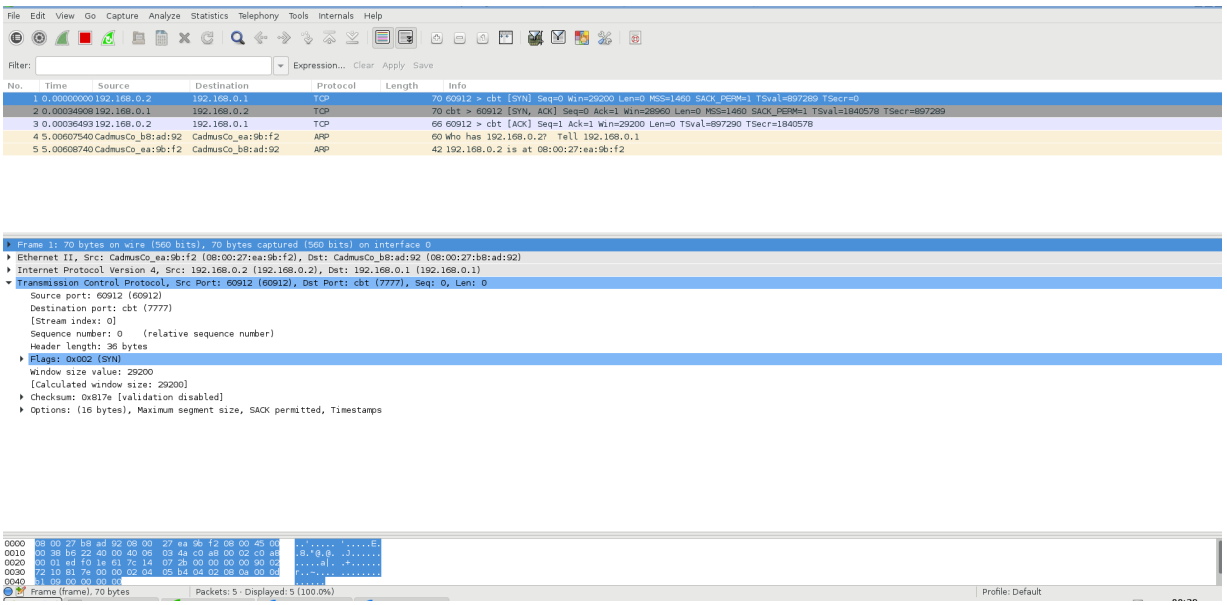
Ejercicio 10. Iniciar una captura de Wireshark. Abrir el servidor en el puerto 7777 y realizar una conexión desde la VM cliente. Estudiar el valor de las opciones que se intercambian durante la conexión. Variar algunos de los parámetros anteriores (ej. no usar ACKs selectivos) y observar el resultado en una nueva conexión.

Adjuntar una captura de pantalla de Wireshark donde se muestren las opciones TCP.

Como es por defecto:

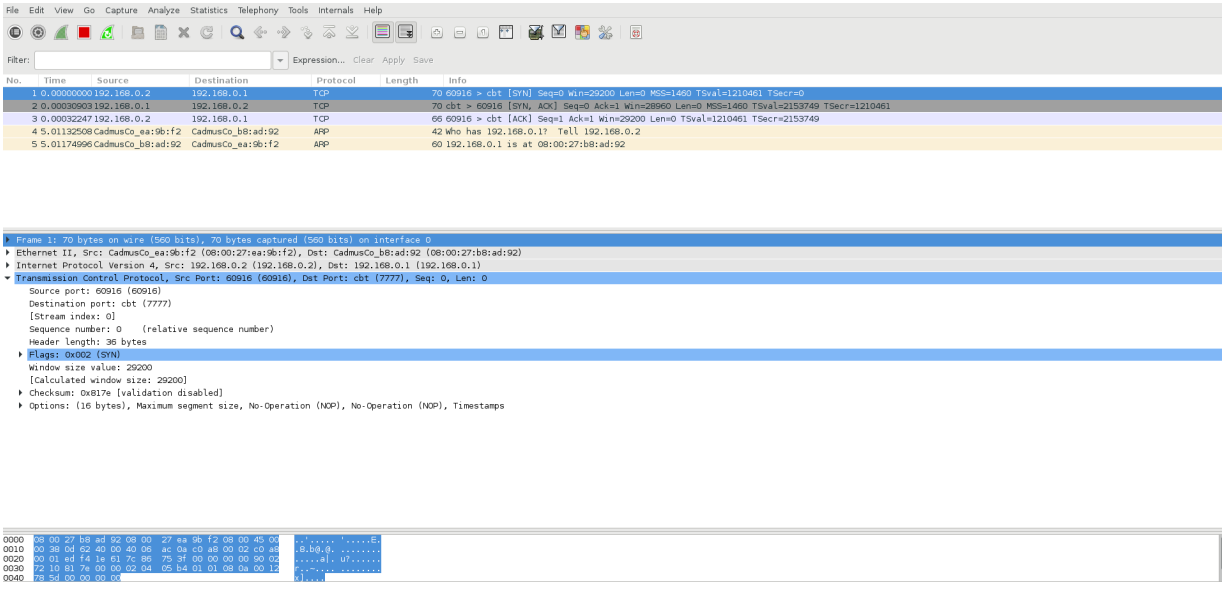


*Hacemos **sysctl net.ipv4.tcp_window_scaling**=0 en ambos servidor y cliente:*



Vemos que ha cambiado la longitud de la cabecera de 40 a 36, las opciones de 20 a 16 y desaparecen las opciones NOP y window scale

*Hacemos **sysctl net.ipv4.tcp_sack**=0 en ambos servidor y cliente:*



Vemos que desaparece la opción Sack permitted, y NOP aparece por duplicado

Ejercicio 11. Con ayuda del comando `sysctl` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten configurar el temporizador *keepalive*:

Parámetro del kernel	**Propósito**	**Valor por defecto**
`net.ipv4.tcp_keepalive_time`	Los procesos de keepalive TCP esperan 2 horas (7200s) la actividad del socket antes de enviar el primer keepalive probe	7200
`net.ipv4.tcp_keepalive_probes`	Número de mensajes keepalive que se envían antes de determinar si se debe cerrar el socket	9

`net.ipv4.tcp_keepalive_intvl`	Intervalo de reenvío de los mensajes keepalive	75

Traducción de direcciones (NAT) y reenvío de puertos (*port forwarding*)

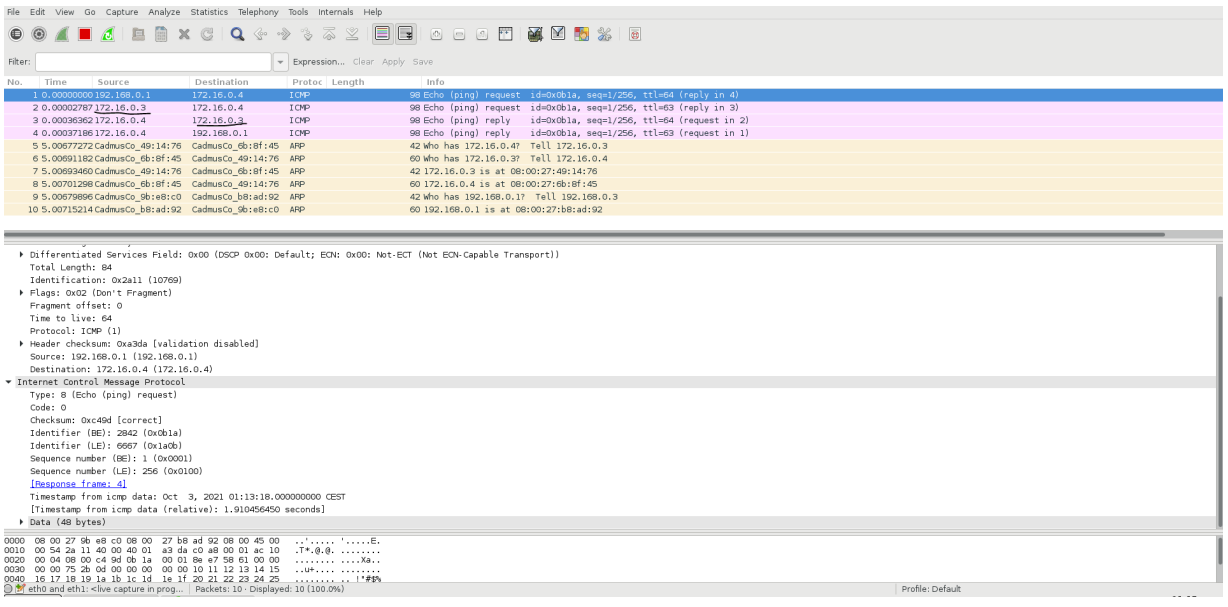
En esta sección supondremos que la red que conecta Router con VM4 es pública y que no puede encaminar el tráfico `192.168.0.0/24`. Además, asumiremos que la dirección IP de Router es dinámica.

Ejercicio 12. Configurar la traducción de direcciones dinámica en Router:

- (Router) Usando `iptables`, configurar Router para que haga SNAT (*masquerade*) sobre la interfaz `eth1`. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Comprobar la conexión con VM4 usando la orden `ping`.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes

Adjuntar el comando iptables utilizado y capturas de pantalla de Wireshark.
*En **VM3** escribimos : iptables -t nat -A POSTURING -o eth1 -j MASQUERADE*
*En **VM1**: ping -c 1 172.16.0.4*
PING 172.16.0.4 (172.16.0.4) 56(84) bytes of data.
64 bytes from 172.16.0.4: icmp_seq=1 ttl=63 time=0.737 ms

--- 172.16.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.737/0.737/0.737/0.000 ms



Se puede apreciar que eth1(2º y 3º mensajes) la dirección de VM1 viene enmascarada por la de VM3

Ejercicio 13. Comprueba la salida del comando `conntrack -L` o, alternativamente, el contenido del fichero `/proc/net/nf_conntrack` en Router mientras se ejecuta el `ping` del ejercicio anterior. ¿Qué parámetro se utiliza, en lugar del puerto origen, para relacionar las solicitudes con las respuestas?

Adjuntar la salida del comando conntrack y responder a la pregunta.

En VM3(Router) : justo al hacer el ping en VM1

icmp 1 28 src=192.168.0.1 dst=172.16.0.4 type=8 code=0 id=2898 src=172.16.0.4 dst=172.16.0.3 type=0 code=0 id=2898 mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.

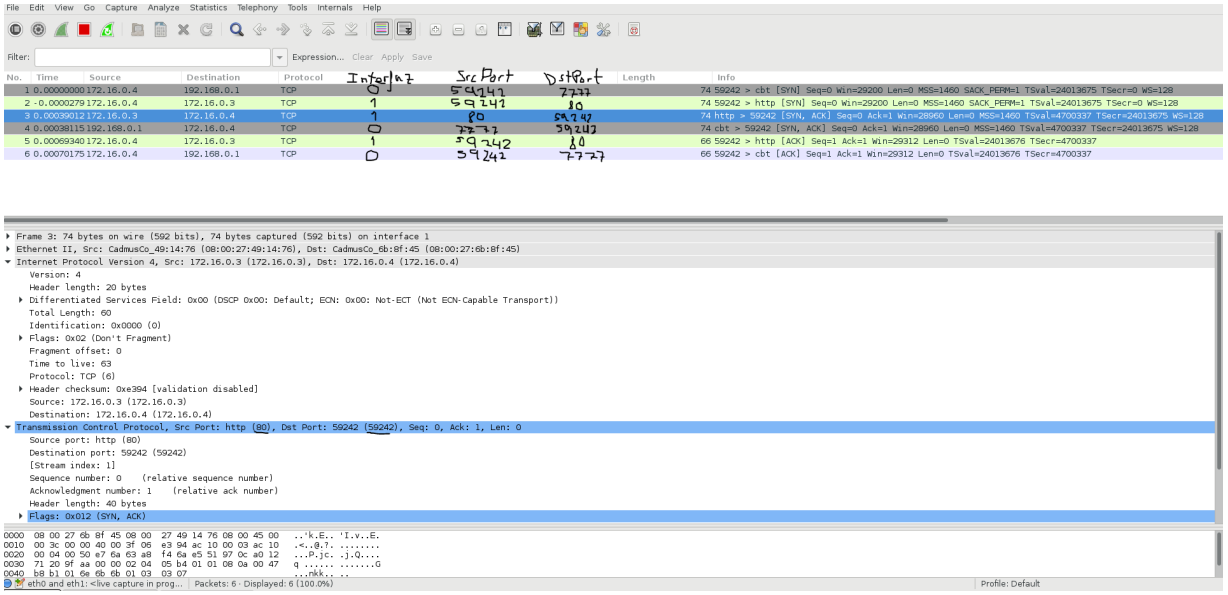
Ejercicio 14. Acceso a un servidor en la red privada:

- (Router) Usando `iptables`, reenviar las conexiones (DNAT) del puerto 80 de Router al puerto 7777 de VM1. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Arrancar el servidor en el puerto 7777 con nc.
- (VM4) Conectarse al puerto 80 de Router con nc y comprobar el resultado en VM1.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes.

Adjuntar el comando iptables utilizado y capturas de pantalla de Wireshark.

*En **VM3**: sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to 192.168.0.1:7777*

*En **VM1**: nc -l 7777 y conectamos en **VM4** con nc 172.16.0.3 80*



Apreciamos que en eth0 aparecen las direcciones ip de VM1 y VM4 192.168.0.1 y 172.16.0.4 respectivamente, y sus puertos 7777 59242; pero en eth1 en vez de lo correspondiente a VM1 aparece lo de VM, dir ip 172.16.0.3 y puerto 80.